|  |
| --- |
|  |

## Краткая история систем кодирования символов естественных языков в США, Европе и Восточно-азиатских странах

## Общая информация о кодировании символов естественных языков

**Кодирование символов: основа для обеспечения возможности работы компьютеров с текстовыми данными**

Важнейшей для большинства людей возможностью компьютерной техники является не то, что эта техника может просто быстро производить математические вычисления - ведь множество людей до сих пор ведут финансовые расчёты, используя микрокалькуляторы, а не ПК, - а то, что компьютерная техника может быть использована для работы с текстовыми данными. Вероятно, каждый должен согласиться с тем, что подавляющее большинство пользователей ПК используют программы для обработки текстов чаще, чем программы какого-либо другого назначения. И с тем, что сейчас сложно будет найти компьютер, на котором не была бы установлена какая-либо программа для обработки текстов.

При этом многие люди, вероятно, и не догадываются о том, что текстовые данные в ПК представлены, в конечном итоге, в виде числовых данных. В современных компьютерных системах символы\*, которые люди используют для записи и передачи текстов на своём языке, кодируются с помощью чисел, записанных в двоичной системе счисления (см. Табл. 1). Это делается потому, что микропроцессоры, лежащие в основе современных компьютерных систем, могут делать, по сути, лишь две вещи: производить действия двоичной арифметики и выполнять Булевы логические операции\*\*.

|  |
| --- |
| \* Под "символами" далее в тексте статьи понимаются и цифры, и буквы, и иероглифы - *прим. перев.* \*\* Только в первом случае последовательность нулей и единиц трактуется как число в двоичной системе счисления, а во втором - как сложное логическое высказывание, в котором "1" обозначает истинность элементарного высказывания, а "0" - его ложность - *прим. ред.* |

|  |
| --- |
| *Таблица 1. Двоичные значения, стоящие за алфавитно-цифровыми символами на экране в 7-битной кодировке ASCII.* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Символ на экране** | **Двоичное значение для его обработки** | **Символ на экране** | **Двоичное значение для его обработки** |
|  | | | |
| 1 | 0110001 | A | 1000001 |
| 2 | 0110010 | B | 1000010 |
| 3 | 0110011 | C | 1000011 |
| 4 | 0110100 | D | 1000100 |
| 5 | 0110101 | E | 1000101 |

Поэтому когда ПК, к примеру, записывает букву "A" на дискету, он не создаёт изображения буквы "A" на магнитном носителе, а записывает на него двоичное число (состоящее из нолей и единиц), соответствующее букве "A" в специальной таблице символов. Вы спросите: но ведь ПК создаёт изображение буквы "A" на дисплее, когда я нажимаю клавишу "A"!? Конечно. Когда Вы нажимаете клавишу с буквой "A", первая вещь, которая при этом происходит - в клавиатуре генерируется число, соответствующее букве "A". ПК использует это число для выборки изображения буквы "A" из файла шрифта, в котором изображение каждой буквы пронумеровано в порядке, определяемом той или иной таблицей символов. Лишь после этого соответствующее изображение появляется на экране. То же самое происходит при печати текста, с той лишь разницей, что изображение буквы "A" оказывается на бумаге, а не на дисплее\*.

|  |
| --- |
| \* Конечно, в данном абзаце всё изложено в сильно упрощённом виде. Более подробно всё это описано в [1]. - *прим. перев.* |

Такая система, использующая а) пронумерованные таблицы символов и б) числа (двоичные коды) для внутреннего представления в компьютерах символов естественных языков, позволяет ПК гибко и эффективно делать такие вещи, которые невозможно было реализовать на простой пишущей машинке. ПК может не только распечатывать текст на бумаге, но позволяет также хранить, модифицировать, сортировать, принимать и передавать текстовые данные с умопомрачительно высокой скоростью. Наконец, при наличии соответствующего ПО ПК может быть использован для работы с текстами не на каком-то строго заданном одном, а сразу на многих языках\*. Единственная возникающая при этом сложность состоит в том, что при работе с языками, в которых используется большое количество символов, например, с японским, для нумерации этих символов требуется использовать бóльшие числовые значения - и, соответственно, более длинные двоичные коды\*\*.

|  |
| --- |
| \* Поддержка такой функции далее в тексте называется "многоязыковой поддержкой" - *прим. перев.* \*\* При этом необходимо учитывать, что длина двоичного кода зависит от величины представляемого им числа как log2(x). Т.е. для кодирования только букв английского языка (26 символов) достаточно было бы длины кода, равной log2(26)~=5 бит, а для кодирования иероглифов японского языка (65.000 символов) необходимо уже log2(65000)~=16 бит, см. ниже. - *прим. ред.* |

ПК для работы с английским языком, используемые в США , используют 7-битную таблицу символов и метод кодирования ASCII (American Standard Code for Information Interchange)\*, которые обеспечивают возможность кодирования 128 символов (поскольку бит может иметь только два значения - "1" или "0" - используя 7 бит, можно представить максимум 27=128 разных чисел. Набор символов\*\*, использованный для заполнения этой таблицы, включает в себя заглавные и строчные буквы латинского\*\*\* алфавита, арабские цифры, знаки препинания и некоторые управляющие двоичные последовательности. Если используется 8-й бит, который изначально был зарезервирован для контроля чётности (вид проверки корректности передачи данных), метод кодирования ASCII становится 8-битным, или 1-байтным\*\*\*\*. Если же этот 8-й бит используется, как и 7 других, для нумерации символов, а не для контроля чётности, то максимально возможный объём таблицы символов, используемой вместе с методом кодирования ASCII , возрастает до 28=256 позиций.

|  |
| --- |
| \* Далее в тексте совокупность таблицы символов и метода, использованного для кодирования этих символов (чаще - их номеров), которые (таблица и метод) используются вместе, называется "системой кодирования". Как правило, и таблица символов, и метод кодирования, составляющие ту или иную систему кодирования, бывают определены в рамках того или иного стандарта. Система кодирования ASCII является американским официальным государственным стандартом - *прим. перев.* \*\* Обратите внимание: "таблица символов" и "набор символов" - это разные термины. Набор символов называют алфавитом кода. Таблица отличается от набора тем, что в ней символы пронумерованы (или как-либо иначе структурированы). - *прим. перев.* \*\*\* На самом деле, настоящий латинский алфавит и алфавит английского языка не эквивалентны: в латинском алфавите 25 букв, а в английском - 26 (добавлена "w"). В оригинале автор статьи, помимо того, что путается в указанных выше терминах, вдобавок под "латинским" в большинстве случаев подразумевает 26-буквенный алфавит, т.е. не латинский, а именно английский. - *прим. перев.* \*\*\*\* 8 бит, как известно, = 1 байт. С учётом сказанного в абзаце, думаю, понятно, почему далее по возможности используются термины "n-битный" вместо "n-байтный": 7 бит нельзя выразить через байты - *прим. перев.* |

В случае работы с языками типа японского, в которых используется огромное количество символов - десятки тысяч, - для нумерации символов, используемых в этом языке, необходимо использовать числа длиною как минимум в 16 бит (2 байта). 16-битный метод кодирования обеспечивает 216=65.536 возможных позиций в таблице символов.

Однако в стандартную таблицу символов, используемую в настоящее время в японских ПК и определённую, в частности, в японском промышленном стандарте JIS [Japan Industrial Standard] X 0208-1990, внесено только 6879 символов. Этого достаточно для решения многих задач по обработке текстов, встречающихся в повседневной жизни, но мало для написания имён людей, географических названий мест, цитирования исторических данных и даже для написания названий рыб, что японцы едят в ресторанах "суши"!

Кроме этой, необходимо отметить ещё одну проблему. Современные компьютеры могут наиболее эффективно работать с порциями данных, размер которых кратен 8 битам (1 байту). Это потому, что в их электронных схемах данные обычно передаются и обрабатываются порциями по 8, 16, 32 или 64 бита. По этой причине 10-битный или 15-битный метод кодирования окажется неудобным и неэффективным для применения в ПК. С другой стороны, если для нумерации символов использовать слишком длинные числа (двоичные последовательности), объём данных, с которыми придётся работать компьютеру, резко возрастёт, что вновь приведёт к существенному снижению эффективности работы. Например, если для нумерации символов использовать 24 бита, это позволило бы работать с символами всех естественных языков мира, как известных из истории, так и используемых сейчас (224=16.777.216). Однако для нумерации символов большинства языков, имеющих алфавиты, всё же достаточно чисел длиною в 8 бит, поскольку в этих алфавитах насчитывается не более 256 символов. И поэтому использование для их представления 24-битного метода кодирования привело бы к обработке 16-ти ненужных в данном случае бит, и при записи той же латинской буквы "А" на дискету привело бы к нерациональному расходованию места на носителе, которое могло бы быть использовано для хранения данных.

## Ранняя история кодирования символов естественных языков

**Телеграфия и начало электронной обработки данных**

Каждый, кто когда-нибудь смотрел кино об американском Западе XIX-го века, знает, что первым методом кодирования, широко применяемым для преобразования символов и текстов в электронный вид, был метод, предложенный Морзе\*. То, что этот метод кодирования был изобретён для передачи сообщений по телеграфным линиям, а не для обработки текстов в компьютерах, общеизвестно. Однако гораздо меньшее количество людей знает, что его изобретатель, американец Сэмюэл Финли Бриз Морзе [Samuel Finley Breese Morse] (1791-1872), был также известным художником. В своё время он изучал рисование в Лондоне и там узнал об исследованиях электромагнетизма, проводимых британскими учёными. Возвращаясь по морю в США в 1832-м году, он задумал создать свою собственную систему телеграфа. Именно с этой системы началось движение в сторону того мира электронных сетей, в котором мы сейчас живём, и именно за её разработку Морзе снискал славу "американского Леонардо да Винчи".

|  |
| --- |
| \* Более устоявшимися названиями являются "код Морзе" или "азбука Морзе", однако они терминологически неточны. В частности, слово "код" имеет несколько значений - помимо значения "метод кодирования", оно может быть интерпретировано и как "текст программы", как "номер символа в таблице символов", и даже как "ключ дешифрации". Что касается термина "азбука Морзе", то им обозначается набор символов, который используется вместе с методом кодирования, предложенным Морзе. Поэтому далее в тексте статьи эта "официальная терминология", приводящая к путанице, не используется - *прим. перев.*) |

Морзе не был первым изобретателем системы телеграфа для практического применения. Эта честь принадлежит двум британцам\*: сэру Чарльзу Уитстоуну [Charles Wheatstone], физику и изобретателю, и сэру Уильяму Фосерджиллу Куку [William Fothergill Cooke], электроинженеру, которые внедрили первую систему железнодорожного телеграфа в Англии в 1837-м году, в то время как Морзе в это время только изобрёл первый американский телеграф. Однако, система британцев была весьма непростой. В ней использовалось 5 проводов, которые отклоняли магнитную иглу-стрелку в приёмнике так, что она указывала на те или иные буквы алфавита\*\*. В отличие от неё, система Морзе была проще. В ней использовался один провод для передачи сигнала, и вместо отклоняемой магнитной стрелки в ней использовался электромагнит, который притягивал небольшую конструкцию, когда поступал сигнал. Это сделало устройство Морзе более надёжным. Другой особенностью изобретения Морзе была возможность записи поступающих сигналов на движущуюся бумажную ленту (так как конструкцию, которая притягивалась электромагнитом, можно было "заставить" перфорировать или окрашивать при этом ленту); тем не менее, на протяжении многих лет эта возможность не использовалась, и люди-операторы были вынуждены читать приходящие сообщения по щелчкам приёмника. Морзе продемонстрировал свою систему кодирования 24 мая 1844-го года в первом в истории США сеансе телеграфной связи, который проводился между городами Балтимор (штат Мэриленд) и Вашингтон (Округ Колумбия). Он послал сообщение "What hath God wrought!" ("Чудесно творение господне!").

|  |
| --- |
| \* Здесь автор ошибается: изобретателем первой в мире работоспособной телеграфной машины - для передачи сообщений на расстояние с помощью электрического тока - был русский инженер Павел Львович Шиллинг. Он построил комплекс устройств для телеграфной связи ещё в 1832-м году - *прим. перев.*  \*\* Буквы наносились на специальную шкалу - *прим. перев.* |

Морзе изобрёл метод кодирования, который он использовал для посылки своего исторического сообщения, в 1838-м году. Метод кодирования Морзе напоминает двоичный код, используемый в современных компьютерах, тем, что он тоже базировался на двух возможных значениях - в случае метода кодирования Морзе, это были точка или тире. Однако в отличие от современных методов кодирования, используемых для нумерации символов в современных компьютерах, комбинации точек и тире, используемые для представления символов в методе кодирования Морзе, были разной длины\* (см. Табл. 2). Морзе использовал принцип, по которому наиболее часто употребляемым буквам ставились в соответствие наиболее короткие последовательности из точек и тире, что существенно сокращало длину сообщения. Например, наиболее часто используемой в английском языке букве "E" в методе кодирования Морзе соответствует одна точка; второй по частоте использования букве английского языка - "T" - одно тире. Интересно, что Морзе подсчитывал частоту использования букв не путём изучения текстов, а путём подсчёта литер каждого типа в типографском наборе. Результатом его поистине каторжного труда стал высокоэффективный метод кодирования, который с некоторыми изменениями используется до сих пор, хотя с момента его изобретения прошло уже более 160 лет.

|  |
| --- |
| \* Здесь следует сделать небольшое пояснение. В компьютерной технике, в случае использования, например, 8-битного метода кодирования для представления символов, номер каждого символа в кодовой таблице всегда задастся числом длиною в 8 бит, даже если для записи этого числа достаточно использовать меньшее число бит. Например, символу в 3-й по счёту позиции в таблице символов в случае использования 8-битного метода кодирования будет соответствует двоичная последовательность не "10", а "00000010". Символу в последней, 256-й, позиции этой таблицы будет соответствовать последовательность "11111111". Собственно, это показано автором статьи в Табл. 1, но внимание на этом заострено не было. - *прим. перев.* |

|  |
| --- |
| *Таблица 2. Подмножество Международной системы кодирования Морзе.* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | .- | N | -. | 0 | ----- |
| B | -... | O | --- | 1 | .---- |
| C | -.-. | P | .--. | 2 | ..--- |
| D | -.. | Q | --.- | 3 | ...-- |
| E | . | R | .-. | 4 | ....- |
| F | ..-. | S | ... | 5 | ..... |
| G | --. | T | - | 6 | -.... |
| H | .... | U | ..- | 7 | --... |
| I | .. | V | ...- | 8 | ---.. |
| J | .--- | W | .-- | 9 | ----. |
| K | -.- | X | -..- | , | --..-- |
| L | .-.. | Y | -.-- | . | -.-.-. |
| M | -- | Z | --.. | ? | ..--.. |
|  | | | | | |

Система кодирования Морзе с течением времени эволюционировала и существовала в нескольких версиях. Всё началось с "Ранней системы кодирования Морзе" [Early Morse Code] , которая впоследствии развилась в "Американскую систему кодирования Морзе" [American Morse Code], а затем - в "Международную систему кодирования Морзе" [International Morse Code]. В Табл. 2 читатель может видеть подмножество символов, являющееся извлечением из таблицы символов Международной системы кодирования Морзе. Заметим, что в указанном подмножестве нет строчных букв, а все арабские цифры представлены с помощью совокупностей, состоящих из 5-ти точек или тире\*. При реальной передаче данных сигнал для тире в 3 раза превосходит по длительности сигнал для точки. Сигналы точек и тире в совокупностях, которыми обозначаются буквы, разделяются интервалами, длительность которых равна длительности сигнала точки. "Пробел" между буквами, формирующими то или иное слово, обозначается интервалом, который по длительности равен утроенной длительности сигнала точки (иными словами, длительность этого интервала равна длительности сигнала для тире). Пробел между словами обозначается интервалом, по длительности равным ушестерённой длительности сигнала точки (иными словами, длительность этого интервала равна удвоенной длительности сигнала для тире).

|  |
| --- |
| \* Следует заметить, что символы других языков (не английского) в Международной системе кодирования Морзе представляются с помощью той же самой "азбуки" (что продемонстрировано в Табл. 2). Таким образом, прочитать сообщение, записанное с использованием системы Морзе, можно только в том случае, если заранее известно, на каком языке оно написано. Далее в статье автор развивает эту тему - *прим. перев.* |

После того, как изобретения Морзе вошли в практическое употребление, другие изобретатели внесли свой вклад в развитие телеграфа, разработав, например, такие устройства, как реле, которые позволяли одновременно с приёмом сигнала в одном месте посылать его далее. Кроме того, были разработаны различные методы, позволяющие более полно использовать пропускную способность телеграфных линий. Так появилось диплексирование [diplexing] (посылка по линии одновременно двух сообщений в одном направлении), дуплексирование [duplexing] (посылка по линии одновременно двух сообщений в разных направлениях), квадроплексирование [quadruplexing] (посылка по линии одновременно четырёх сообщений, по два в каждом направлении). Впоследствии приём поступающих сигналов был механизирован с помощью ленточитающих машин (см. выше), производительность которых в 1900-м году составляла порядка 400 слов в минуту.

Следующим существенным прорывом вперёд в технологии телеграфа был примитивный печатающий телеграфный аппарат, или "телетайп"; его запатентовал во Франции в 1874-м году Жан Морис Эмиль Бодо [Jean-Maurice-Émile Baudot] (1845-1903). Как и телеграф Морзе, это изобретение привело к созданию новой системы кодирования символов - 5-битного метода кодирования Бодо (и используемой с ним таблицы символов). Метод кодирования Бодо стал первым в мире методом кодирования текстовых данных с помощью двоичных последовательностей. Сообщения, для передачи которых использовалась система кодирования Бодо, распечатывались операторами на узкие ленты для двухканальной связи с помощью специальных 5-клавишных клавиатур. В более поздних версиях устройства использовались клавиатуры с буквами (как у пишущих машинок - *прим. перев.*), которые уже автоматически генерировали соответствующую той или иной букве 5-битную последовательность. Другой особенностью телетайпа Бодо было мультиплексирование, обеспечивающее возможность одновременной работы до 6-ти операторов благодаря применению системы временнóго распределения. Это позволило значительно увеличить пропускную способность телеграфной линии. Предложенная Бодо аппаратура зарекомендовала себя весьма положительно и оставалась в широком применении в XX-м веке, пока её не вытеснили телефоны и персональные компьютеры.

|  |
| --- |
| *Таблица 3. Кодовая таблица Бодо.* |

|  |  |  |
| --- | --- | --- |
| **Двоичное значение** | **Буквы** | **Знаки** |
|  | | |
| 00011 | A | 1 |
| 11001 | B | ? |
| 01110 | C | : |
| 01001 | D | $ |
| 00001 | E | 3 |
| 01101 | F | ! |
| 11010 | G | & |
| 10100 | H | STOP |
| 00110 | I | 8 |
| 01011 | J | ' |
| 01111 | K | ( |
| 10010 | L | ) |
| 11100 | M | . |
| 01100 | N | , |
| 11000 | O | 9 |
| 10110 | P | 0 |
| 10111 | Q | 1 |
| 01010 | R | 4 |
| 00101 | S | BELL |
| 10000 | T | 5 |
| 00111 | U | 7 |
| 11110 | V | ; |
| 10011 | W | 2 |
| 11101 | X | / |
| 10101 | Y | 6 |
| 10001 | Z | " |
| 00000 | n/a | n/a |
| 01000 | CR | CR |
| 00010 | LF | LF |
| 00100 | SP | SP |
| 11111 | LTRS | LTRS |
| 11011 | FIGS | FIGS |
|  | | | |

Способ кодирования, который использовал Бодо в своей системе, подразумевал использование 5 бит, однако 32-х позиций (25=32) недостаточно для представления символов латинского алфавита вместе с арабскими цифрами и знаками препинания. Поэтому в коде Бодо используется специальная схема "сдвига" для переключения между двумя "внутренними" таблицами символов, по 32 символа в каждой (см. Табл. 3), которую можно сравнить с системой "сдвига", используемой в пишущих машинках для переключения на верхний регистр. Технически это реализовывалось с помощью использования управляющих (в данном случае работой телетайпов) двоичных последовательностей. Местá для подобных управляющих двоичных последовательностей есть и в таблицах символов, которые используются в современных системах кодирования, применяемых в сегодняшних ПК. Как видно из Табл. 3, в таблице символов кода Бодо имеются те же символы, что показаны в приведённом выше извлечении из таблицы символов Международной системы кодирования Морзе: заглавные буквы латинского алфавита, арабские цифры и знаки препинания. В общей сложности печатаемых символов в первоначальной версии таблицы символов кода Бодо (изображённой в Табл. 3) было 50. Составляющие, т.е. "единицы" и "нули", двоичного кода в аппарате Бодо обозначались пометками и пробелами на ленте; во время передачи сообщения они использовались для управления включением и выключением электрического тока. Каждый символ в сообщении предварялся стартовым битом, а после символа передавался стоповый бит, что сделало метод кодирования Бодо медленным по нынешним меркам.

Причиной, по которой Бодо был вынужден ограничить свой метод кодирования использованием двоичных последовательностей длиною в 5 бит и не оставил, таким образом, места в таблице символов для строчных латинских букв - были аппаратные ограничения. Более сложный метод кодирования - даже использовавший бы для представления символов всего на 1 бит больше (т.е. 6-битный) - потребовал бы применения значительно более сложных электромеханических передающих устройств, изготовление (по крайней мере массовое изготовление - *прим. перев.*) которых во времена Бодо было чрезвычайно трудной технологической задачей. После расширения системы кодирования Бодо так, что с её помощью стало возможным передавать 55 разных символов - и при этом появились 3 позиции для "символов национальных языков" - CCITT в Женеве, в Швейцарии, в 1932-м году объявил её стандартной 5-битной системой кодирования символов для телетайпов. Она получила обозначение "Международный телеграфный алфавит No.2". (что терминологически неправильно - *прим. перев.*). В честь Бодо, между прочим, была также названа специальная единица измерения Бод (Baud), которой измеряется число изменений (модуляций) сигнала в секунду\*.

|  |
| --- |
| \* При этом нужно отметить, что, как сказано в [2], "в зависимости от используемого способа модуляции каждое изменение сигнала может соответствовать не только одному, но и большему количеству бит"; таким образом "бод" и "бит в секунду", вопреки распространённому ошибочному представлению - это разные вещи. - *прим. перев.* |

**1890-й год: американское Бюро статистики и появление в нём новой системы для кодирования символов**

Конец XIX-го века ознаменовался изобретением ещё одной системы для кодирования символов естественных языков. Он была создана в США для обработки данных, собираемых во время переписей населения, молодым американским\* изобретателем, которого звали Герман Холлерит (1860-1929). После окончания Нью-Йоркской Колумбийской школы горного дела в 1879-м году, в 1880-м Холлерит устроился на работу в Бюро статистики при Министерстве внутренних дел США. Последствия этого, казалось бы, незначительного события сказывались впоследствии вплоть до 1970-х годов, "золотой эры" больших вычислительных машин. Герман Холлерит был ни кем иным, как изобретателем кода Холлерита, использовавшегося для записи алфавитно-цифровой информации (т.е. символов английского алфавита, используемых в нём знаков препинания, а также арабских цифр - *прим. перев.*) на бумажных перфорированных картах. С появлением перфокарт в американских массах возникло шуточное выражение 'do not fold, spindle, or mutilate' ("не гнуть, не тянуть, не мять"), которое оставляло у многих американцев впечатление, что компьютеры скоро будут полностью контролировать их общество.

|  |
| --- |
| \* Холлерит на самом деле не был американцем, его родители - чистокровные немцы, эмигрировавшие в США; подробнее о Холлерите, Бэббидже, Жаккаре (см. ниже) и их изобретениях см. в [3] - *прим. перев.* |

Интересно, что хотя Холлерит применил перфокарты в работе Бюро статистики в 1880-х, изобретены они были гораздо раньше. Первые перфокарты были изготовлены около 1800-го года французским изобретателем Жозефом Мари Жаккаром (1752-1834), который использовал их в новых фабричных ткацких станках. Это изобретение, запатентованное Жаккаром в 1804-м году, позднее позволило полностью автоматизировать работу ткацких станков, в результате чего текстильная промышленность стала такой, какой мы её знаем сегодня. Более того, Холлерит не был первым, кто предложил использовать изобретение Жаккара для использования в вычислительной технике. Эти лавры принадлежат англичанину Чарльзу Бэббиджу (1792-1871), который планировал использовать перфокарты в своей "аналитической машине", огромном механическом компьютере, который так никогда и не был полностью построен (при жизни Бэббиджа; позднее эту работу завершил его сын - *прим. перев.*). Наконец, Холлерит и не предложил бы использовать перфокарты для ввода данных, если бы не завершились неудачно его эксперименты с другим носителем - перфорированной бумажной лентой, которая имела склонность разрываться во время работы.

Чтобы решить проблему с разрывами бумажной ленты, Холлерит изготовил то, что впоследствии стало компьютерной перфокартой ([по этому адресу](http://www.loc.gov/exhibits/treasures/images/0009.jpg) доступно фото оригинальной перфокарты Холлерита, хранимой в коллекции библиотеки Конгресса США). В том виде, в каком она была позднее стандартизирована, перфокарта была жёсткой карточкой (из плотной бумаги) размером 186x82 мм со срезанным верхним левым уголком. На ней были 12 горизонтальных рядов (пронумерованных сверху вниз как 12, 11, 10, 1, 2, 3, 4, 5, 6, 7, 8, 9) и 80 вертикальных (пронумерованных слева направо с 1-го по 80-й включительно). Каждый вертикальный ряд соответствовал одному алфавитно-цифровому символу. Когда перфокарты протягивались через специальное считывающее устройство, называемое табулятором, гибкие игольчатые контакты попадали в соответствующие дырки, замыкая одну из электрических цепей; таким образом регистрировалось то или иное значение. Несмотря на то, что эта система выглядела очень функционально ограниченной и примитивной, на самом деле она была очень мощной и эффективной. Перепись населения США в 1880-м году, которая выполнялась ручным трудом, продолжалась 7 лет, и полученные данные за это время стали, естественно, устаревшими и бесполезными. Когда при переписи населения в 1890-м году были впервые использованы табуляторы Холлерита, Бюро статистики с их помощью провело все необходимые вычисления за 6 недель и с более точными данными, чем ранее, что позволило сэкономить 5.000.000 долларов США. С тех пор стоимость работ, проводимых Бюро статистики, существенно упала.

В системе кодирования Холлерита, на первый взгляд, двоичная последовательность, соответствующая одному алфавитно-цифровому символу, регистрировалась в 12-ти горизонтальных рядах перфокарты, и поскольку каждый из рядов мог быть перфорирован или не перфорирован (содержать 1 или 0), это давало бы 12-битный метод кодирования. Максимально возможное количество элементов, которое могло бы быть представлено с помощью такого метода кодирования - 212, то есть 4096. На самом же деле таблица символов, используемая в системе Холлерита, содержала только 69 символов - это были заглавные латинские буквы, арабские цифры, знаки препинания и некоторые другие символы. Таким образом, реальные возможности системы кодирования Холлерита по представлению текстовых данных были сравнимы даже с таковыми возможностями системы кодирования Бодо. Зачем же тогда, спросит читатель, на перфокарте было предусмотрено так много горизонтальных рядов? Одной из причин этому является тот факт, что большее количество рядов позволяет кодировать данные, производя меньшее количество перфораций, а это было критично, так как работа с перфораторами карт в течение многих лет производилась вручную. Поскольку рядов было много, для кодирования каждой из 10 арабских цифр (а также 2-х букв латинского алфавита) было достаточно сделать всего одну перфорацию на один вертикальный ряд. Оставшиеся 24 буквы латинского алфавита кодировались с помощью двух перфораций. Для кодирования знаков препинания и прочих символов требовалось сделать 2 или больше перфораций, поэтому нередко они (для экономии времени - *прим. перев.*) вообще не использовались.

Поскольку использование табулятора в Бюро статистики оказалось столь успешным, Холлерит решил начать с его помощью своё дело. В 1896-м году он основал компанию для продажи своего изобретения, которая, как несложно догадаться, была названа по названию выпускаемого устройства - "Tabulating Machine Co.". В 1911-м году она слилась с двумя другими компаниями - "Computing Scale Co. of America" и "International Time Recording Co.", после чего получила название "Computing-Tabulating-Recording Co.", или, сокращённо, C-T-R. После захвата рынков на нескольких континентах и организации деловых операций в общемировом масштабе, в 1924-м году C-T-R сменила своё название на "International Business Machines Corporation" - IBM. И поскольку система кодирования Холлерита и перфокарты были основными технологиями этой корпорации - доминирующего производителя на рынке компьютерной техники в 1960-х - 1970-х гг. - они удостоились широкого использования для представления данных в компьютерных системах и применялись вплоть до 1960-х, пока IBM не разработала иной код для своих больших вычислительных машин.

## Использовавшиеся в компьютерах системы кодирования символов американского английского языка и символов европейских языков

**Системы кодирования ASCII, ISO 646 и EBCDIC**

Быстрое развитие коммуникационных средств и технологий для обработки данных в первой половине XX-го века в США сделало очевидной необходимость в создании стандартной системы кодирования для обмена текстовой информацией. Эта система должна была обеспечивать представление всего того набора символов, что имеется в англоязычной пишущей машинке. Американская стандартизирующая ассоциация - 'American Standards Association', ASA, которая позднее сменила название на 'American National Standards Institute', ANSI - занялась изучением этой проблемы в конце 1950-х гг. и в конце концов пришла к заключению, что система кодирования, в которой используется 7-битный метод кодирования - когда для представления каждого из символов используется двоичная последовательность длиною в 7 бит, - устраняет необходимость в "сдвиге", используемом в системе кодирования Бодо. Поэтому использования 7-битного метода кодирования будет достаточно для достижения поставленной цели. В 1963-м году ASA предложила систему кодирования ASCII (American Standard Code for Information Interchange), которая изначально называлась, очевидно, ANSCII (American National Standard Code for Information Interchange). В любом случае, в таблице символов, используемой в системе кодирования ASCII, в том виде, в каком она была заявлена в 1963-м, оставалось много зарезервированных позиций, в том числе для строчных латинских букв, которые на тот момент были пустыми. Только в 1968-м году они были заполнены, и тогда система кодирования ASCII, в таблице символов которой было 128 позиций - для 32-х управляющих двоичных последовательностей и 96-ти печатаемых символов, - приняла свой окончательный вид, в котором она используется в настоящее время (см. Табл. 4. Заметим, что позиции для пробела [space] и для удаления [delete] оставлены пустыми; см. также Рис. 5 ниже). Более того, несмотря на то, что система кодирования ASCII специально разрабатывалась так, чтобы избежать необходимости в "сдвиге", в её таблицу символов были включены управляющие двоичные последовательности для его обеспечения - в частности, SI (SHIFT IN ) и SO (SHIFT OUT ) для "фиксированного сдвига", а также ES (ESCAPE ) для "нефиксированного сдвига"\*. Эти управляющие двоичные последовательности были позднее использованы для расширения системы кодирования ASCII таким же образом, как это реализовано в системе кодирования Бодо - две "внутренние" таблицы символов объёмом в 128 позиций каждая составляли одну таблицу символов объёмом в 256 позиций, в которой для кодирования доступных для печати символов было отведено 190 позиций\*\*.

|  |
| --- |
| \* Люди, работавшие с пишущей машинкой, конечно, знают, что там набор на верхнем регистре может быть фиксирован - например, для печати заголовков - и снят нажатием специальной клавиши. Очевидно, управляющий символ SI "эквивалентен" нажатию клавиши фиксатора, SO - повторному её нажатию, то есть снятию с фиксатора. ES, очевидно, можно сравнить с нажатием и удержанием в течение некоторого времени клавиши перевода на верхний регистр без использования фиксатора (т.е. для печати одной, нажимаемой сразу после этого, "буквы" на верхнем регистре). Хотя в тексте статьи автор по этому поводу сам себе противоречит, см. ниже. - *прим. перев.*  \*\* Возможно, когда-то это так и было, но очень давно. Используемые сейчас "расширения системы кодирования ASCII" используют не 7, а всё же 8 бит для представления символов. Другое дело, что в самих этих "расширениях системы кодирования ASCII", 8-битных по своей сути, escape-последовательности используются иногда описанным образом для дальнейшего расширения таблицы символов. Об этом в статье говорится ниже. - *прим. перев.* |

|  |
| --- |
| *Таблица 4. Фрагмент кодовой таблицы ASCII (94 позиции).* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | @ | P | ` | p |
| ! | 1 | A | Q | a | q |
| " | 2 | B | R | b | r |
| # | 3 | C | S | c | s |
| $ | 4 | D | T | d | t |
| % | 5 | E | U | e | u |
| & | 6 | F | V | f | v |
| ' | 7 | G | W | g | w |
| ( | 8 | H | X | h | x |
| ) | 9 | I | Y | i | y |
| \* | : | J | Z | j | z |
| + | ; | K | [ | k | { |
| , | < | L | \ | l | | |
| - | = | M | ] | m | } |
| . | > | N | ^ | n | ~ |
| / | ? | O | \_ | o |  |

Система кодирования ASCII была принята всеми изготовителями компьютеров в США за исключением корпорации IBM, которая разработала собственную "фирменную" систему кодирования символов для использования в своих больших ЭВМ (см. ниже). А поскольку американские производители компьютеров были в то время крупнейшими поставщиками компьютеров в мире, система кодирования ASCII немедленно стала международным стандартом de facto. Это вызвало необходимость адаптировать систему кодирования ASCII для других (кроме английского) языков, использующих латинский алфавит, в частности, для языков западноевропейских государств. Эта работа была проделана Международной организацией по стандартизации (International Organization for Standardization, ISO), базирующейся в Женеве, Швейцария. В 1967-м году была выпущена рекомендация ISO 646. Она, в основном, ссылалась на использование кода ASCII "как есть", за исключением того, что 10 позиций в таблице символов должны быть оставлены открытыми для "национальных вариантов", как в коде Бодо (в нём, как мы помним, таких позиций было 3). Десять символов, которые должны были занимать эти позиции в кодовой таблице, были указаны в версии этой рекомендации, известной как International Reference Version (IRV). Таблица символов, используемая в системе кодирования ASCII, была также использована в качестве базовой при создании 7-битных таблиц символов для языков, не использующих латинский алфавит, например, арабского и греческого\*. В 1969-м году она была сделана основой таблицы символов для японского языка, определённой стандартом JIS. На данный момент существует в общей сложности порядка 180 таблиц символов для различных языков, используемых в 8-битных системах кодирования, называемых "расширениями системы кодирования ASCII", зарегистрированных ISO.

|  |
| --- |
| \* Необходимо вновь сделать пояснение. В современных "расширениях системы кодирования ASCII" для представления символов используется, как уже говорилось, 8-битный метод кодирования (т.е. при записи на носитель одному символу всегда соответствует двоичная последовательность не в 7, а в 8 бит длиной). Однако это не мешает рассматривать используемую в таких "расширениях" таблицу символов как состоящую из двух "половинок" по 128 символов в каждой. Можно сказать, из двух 7-битных "половинок", т.к. для представления символов в каждой из которых достаточно использовать 7 бит. Восьмой бит, используемый в методе кодирования, таким образом, можно рассматривать как бит, который определяет, какая из двух "половинок" используется. Можно это продолжить, и рассматривать как "определяющие номер половинки" уже не 1, а 2 бита. Тогда таблицу символов можно рассматривать как состоящую из 4-х 6-битных "половинок" (точнее, уже "четвертинок") по 64 символа в каждой. В любом случае, как бы мы не рассматривали структуру таблицы, на метод кодирования - то есть на длину используемых для записи одного символа двоичных последовательностей - это никак не влияет. Исторически, однако, сложилось, что 256-позиционные таблицы символов, которые используются в 8-битных системах кодирования, называемых "расширениями системы кодирования ASCII", рассматриваются как состоящие из двух 7-битных "половинок". Это можно понять с учётом того, что оригинальная система кодирования ASCII была 7-битной. Во всех "расширениях системы кодирования ASCII" в целях совместимости первая 7-битная "половинка" таблицы символов (с позициями с номерами от 0 до 127) соответствует (за малыми исключениями, см. ниже) полной таблице символов, использованной в оригинальной системе кодирования ASCII. А позиции во второй "половинке" (с позициями 128-255) каждой из таблиц символов, используемых в "расширениях системы кодирования ASCII", отводятся под символы того или иного языка - арабского, греческого и т.д. Вот почему 7-битная система кодирования ASCII является основой для этих 8-битных систем кодирования, называемых "расширениями системы кодирования ASCII". - *прим. перев.* |

Конечно, подавляющее большинство людей, соприкасающихся с системой кодирования ASCII, на самом деле работают с ней не в оригинальном виде, а с одной из более поздних её модификаций. Обычно это одно из 8-битных "расширений системы кодирования ASCII", разработанных производителем компьютера\* обычно для использования в каком-либо регионе, а не какой-то одной стране. Англоязычные версии двух ведущих ОС для ПК - Microsoft Windows и Apple Macintosh (MacOS) - поддерживают работу со специальными символами и символами букв с ударениями и диакритическими (надбуквенными - *прим. перев.*) знаками, для работы с европейскими языками, базированными на латинском алфавите (например, с французским, немецким, шведским) без необходимости внесения каких-либо модификаций в ОС или ПО для обработки текстов. При этом специальные символы и символы букв с диакритическими знаками для этих языков вводятся после нажатия одной из функциональных клавиш (или их комбинации - *прим. перев.*), что, к сожалению, снижает скорость печати. Кроме того, следует особо отметить, что (вторые "половинки" - *прим. перев.*) таблицы символов, используемые в этих "расширениях системы кодирования ASCII", заведомо неполны. Например, в них нередко отсутствуют символы денежных единиц, используемых в странах соответствующего региона. Более того, в них "потеряны" некоторые символы, необходимые при обработке исторических текстов и не принимаются во внимание некоторые часто используемые лигатуры - когда две буквы соединены и используются в качестве одной.

|  |
| --- |
| \*Точнее, производителем таблицы знакогенератора для текстового режима работы видеокарты и, также, производителем используемого на компьютере ПО. - *прим. перев.* |

Несмотря на то, что система кодирования ASCII (и различные её "расширения") была принята практически всеми американскими производителями мини-компьютеров, рабочих станций и ПК, и позднее была возведена ISO в ранг международного стандарта, корпорация IBM - бесспорный лидер в области производства Больших ЭВМ в 1960-х - 1970-х гг. - предпочла пойти своим собственным путём. Это обычно объяснялось желанием сохранить совместимость с предыдущими системами, но многие люди интерпретировали данный факт как проявление маркетинговой стратегии IBM, нацеленной на "запирание" покупателей IBM в рамках ПО и аппаратных средств, произведённых IBM. Как бы то ни было, корпорация создала собственную патентованную 8-битную (как мы помним, 28 даст 256 возможных позиций в таблице символов) систему кодирования, названную EBCDIC (Extended Binary Coded Decimal Interchange Code). Она использовалась в коммерчески удачной серии Больших ЭВМ IBM System/360, которые были хитом продаж в апреле 1964 года. EBCDIC, как видно даже из названия (Extended - расширенный), является, по сути, "расширенной" до использования 8 бит более старой системой кодирования, разработанной IBM - BCDIC (Binary Coded Decimal Interchange Code), в которой использовался 6-битный метод кодирования. Следует отметить, что в некоторых источниках BCDIC называется просто BCD (Binary Coded Decimal) и указывается, что она, в свою очередь, происходит от ещё более ранней, системы кодирования, которая применялась для представления арабских цифр и в которой использовался 4-битный метод кодирования (24 = 16 позиций).

Поскольку IBM продавала свои Большие ЭВМ по всему миру, возникло множество разных версий EBCDIC. Известно о 57 различных "национальных" вариантах EBCDIC, в конечном итоге разработанных и использованных корпорацией в своих системах. И несмотря на то, что была создана International Reference Version (IRV) системы кодирования EBCDIC, базированная на рекомендации ISO 646 и поэтому "совместимая" с системой кодирования ASCII\*, большинство таблиц символов, используемых в разных вариантах кода EBCDIC, даже не содержат всех тех символов, что имеются в таблице символов кода ASCII. Это препятствует нормальному обмену данными между системами, на одной из которых для представления символов используется система кодирования EBCDIC, а на другой - система кодирования ASCII. Более того, получить от IBM детальную документацию по многочисленным версиям EBCDIC было чрезвычайно сложно (что ещё более утвердило мнение о том, что главной целью EBCDIC было удерживать покупателей систем, произведённых IBM, в "паутине" этой корпорации), и международный обмен данными даже между "мэйнфреймами" IBM (использующими разные версии EBCDIC - *прим. перев.*) был сложной задачей для системных операторов, которые поэтому быстро стали лютыми врагами EBCDIC. В сегодняшнем основанном на Internet компьютерном мире, где для представления символов естественных языков используются 7-битная система кодирования ASCII и её 8-битные "расширения", утверждённые ISO, система кодирования EBCDIC служит только замедлению обмена информацией. По этой причине большинство коммерческих организаций, которые успели накопить данные, записанные с использованием EBCDIC и других фирменных систем кодирования, конвертируют их в данные, записанные с использованием системы кодирования ASCII.

|  |
| --- |
| \* На самом деле, порядок размещения символов в таблицах символов, применяемых в системах кодирования ASCII и EBCDIC, совершенно различен, см. ниже - *прим. перев*. |

**Системы кодирования для представления специфических символов европейских языков**

В то время как системы кодирования ASCII и ISO 646, в которых использовался 7-битный метод кодирования, были достаточны для обработки данных на английском языке, в частности данных на современном английском языке, они не подходили для обработки данных, написанных с применением специфических символов, используемых в языках большинства стран Европы. В основном это буквы латинского алфавита с проставленными над ними ударениями и диакритическими знаками. Кроме того, в ряде европейских государств (в частности в Греции) используются свои собственные алфавиты, символы которых отличны от латинских букв. Это же касается и алфавитов ряда неевропейских государств - русской кириллицы\*, арабского письма и израильского иврита. Соответственно, после стандартизации систем кодирования ASCII и ISO 646, стала очевидной необходимость в создании ряда новых систем кодирования - для представления символов, используемых в указанных алфавитах, и удовлетворения, таким образом, потребностей европейских государств по работе с данными. Для этого ISO первоначально создала стандарт ISO 2022, который указывает, как должны быть структурированы и развиты существующие 7-битные и 8-битные системы кодирования. Этот стандарт был позднее применён для создания стандарта ISO 8859-1 (имеющего неофициальное название "Latin-1"). ISO 8859-1 определяет таблицу символов одного из 8-битных "расширений системы кодирования ASCII", которое сегодня широко используется для обмена информацией через Internet в странах Западной Европы.

|  |
| --- |
| \* Здесь у автора явно проявляется "синдром белых медведей на улицах Москвы"... - *прим. ред.* |

|  |
| --- |
| *Таблица 5. Ракладка 8-битной кодовой таблицы.* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | | |  | | --- | | 00 | | 01 | | 02 | | 03 | | 04 | | 05 | | 06 | | 07 | | 08 | | 09 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | |  |  | | --- | --- | |  |  | |  |  | | |  | | --- | | CL  a r e a | | | |  |  | |  | ESC | |  |  | |  |  | |  |  | |  |  | | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | SP |  |  |  |  |  | |  |  |  |  |  |  | |  | |  | | --- | | GL  a r e a | | | | |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  | DEL | | | | | | | |  |  | | --- | --- | |  |  | |  |  | | |  | | --- | | CR  a r e a | | | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | ▓ |  |  |  |  |  | |  |  |  |  |  |  | |  | |  | | --- | | GR  a r e a | | | | |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  | ▓ | | | | | | | |

Стандарт ISO 2022 определяет, среди прочего:

* каким образом осуществляется "раскладка" таблиц символов, используемых в 7-битных и 8-битных системах кодирования (8 вертикальных и 16 горизонтальных рядов для таблиц символов 7-битных систем кодирования; по 16 вертикальных и горизонтальных рядов для таблиц символов 8-битных систем кодирования);
* порядок нумерации рядов в этих таблицах (*для 7-битных систем кодирования:* вертикальные ряды: с 0-го по 7-й, горизонтальные: с 0-го по 15-й; *для 8-битных систем кодирования:* вертикальные ряды: с 0-го по 15-й, горизонтальные ряды: тоже с 0-го по 15-й);
* какова внутренняя структура этих таблиц (таблицы, используемые в 7-битных системах кодирования, разбиваются на области [area] CL и GL, а в 8-битных - на области CL, GL, CR и GR. Области, содержащие в названии C, отведены под управляющие последовательности, а области, содержащие в названии G - под символы);
* специальные зарезервированные ячейки [fixed assignments] в этих таблицах, предназначенные для управляющих символов: "escape " (ESC ; позиция в таблице: горизонтальный ряд 0, вертикальный ряд 11), пробел (SP: 2,0) и "delete " (DEL: 7,15). Номера этих ячеек соответствуют их номерам в таблице символов, используемой в системе кодирования ASCII (табл. 5).

Тем не менее, в отличие от системы кодирования ASCII, в которой управляющий символ "escape" использовался для реализации функций, аналогичных "возврату каретки" пишущей машинки, в ISO 2022 этот символ существует для обеспечения "собственно компьютерных" управляющих функций, например, для использования в так называемых escape-последовательностях, предназначенных для переключения таблиц символов.

Наконец, в стандарте ISO 2022 также определена (англоязычная - *прим. перев.*) терминология, используемая при описании кодов, предназначенных для представления символов естественных языков. Поэтому мы рекомендуем тем, кто хочет стать специалистом в данной области, начать её изучение с данного стандарта.

С тех пор как ISO 2022 занял место международного стандарта, он позволил всем другим кодам для представления символов естественных языков использовать в качестве базы ISO 646. Разработчики сравнивали его с пишущей машинкой, печатающие головки у которой можно менять - этими "печатающими головками" стали 7-битные и 8-битные коды, составленные в соответствии с ISO 2022. Наиболее важное "семейство" этих взаимозаменяемых "печатающих головок" было определено в ISO 8859. Этот документ описывает большой набор из 8-битных кодов, нацеленный на удовлетворение нужд по обработке данных стран Западной и Восточной Европы. Он (набор из 8-битных кодов) выглядит следующим образом:

**ISO 8859**

|

+--Часть 1: *Латинский алфавит No. 1* (Пересмотрен в 1998-м году).

| Набор символов, используемых в языках стран Западной Европы (известный также

| как знаменитый "Latin-1");

|

+--Часть 2: *Латинский алфавит No. 2*

| Набор символов, используемых в языках стран Восточной Европы (сербо-хорватском,

| албанском, венгерском, румынском);

|

+--Часть 3: *Латинский алфавит No. 3*

| Набор символов, используемых в языках стран Южной Европы (в частности в

| мальтийском языке) и в языке эсперанто;

|

+--Часть 4: *Латинский алфавит No. 4* (1998)

| Набор символов, используемых в языках стран Северной Европы (скандинавских);

|

+--Часть 5: *Латинский алфавит и кириллица*;

|

+--Часть 6: *Латинский и арабский алфавиты*;

|

+--Часть 7: *Латинский и греческий алфавиты*;

|

+--Часть 8: *Латинский алфавит и иврит*;

|

+--Часть 9: *Латинский алфавит No. 5*

| Набор символов, используемых в современном турецком языке;

|

+--Часть 10: Латинский алфавит No. 6 (1998)

| Набор символов, используемых в исландском, нордических и балтийских языках;

|

+--Часть 13 (DIS) *Латинский алфавит No. 7*;

|

+--Часть 14 (DIS) *Латинский алфавит No. 8* (символы кельтского языка).

Конечно, каждый знает, что в реальном мире коммерческие структуры не спешат соблюдать стандарты, особенно если это стандарты иностранных государств. И именно так случилось с ISO 8859 \*. ISO 8859 не получил широкого распространения до тех пор, пока не появились персональные компьютеры (ПК). А поскольку ПК были, по сути, изобретением американцев, американская компьютерная индустрия посчитала себя вправе и стала адаптировать стандарты ISO для собственных нужд. Следствием этого стали используемые в ПК ОС, в которых использовались таблицы символов со специфическими для этих ОС "расширениями", которые не были полностью (слово "полностью", на самом деле, правильнее было бы опустить - *прим. перев.*) совместимы с ISO 8859. Хотите увидеть результат? Если Вы используете ОС Windows, взгляните на пункт "Document Encoding " в меню Вашего Web-броузера (в Netscape Navigator он доступен через меню "Options"). Там Вы обнаружите перечень различных таблиц символов, используемых в Windows и MacOS.

К счастью \*\*, разработчики сети Internet создали такие протоколы для передачи данных, которые допускали использование абсолютно любых таблиц символов для представления текстовых данных, - даже будь эта таблица разработана одним человеком (см. нижние пункты в меню "Document Encoding" ) - передаваемых через сеть. Различия этих таблиц символов, таким образом, имеют характер вызывающих раздражение затруднений, но не непреодолимого барьера, для международного обмена данными.

|  |
| --- |
| \* Автор здесь ошибается, так как стандарты ISO являются **международными**. - *прим. перев.*  \*\* С мнением автора по этому поводу можно поспорить... - *прим. перев.* |

## Коды "CJK" для представления символов китайского, японского и корейского языков

(аббревиатура CJK происходит от "Chinese, Japanese, Korean" - *прим. перев.*)   
  
  
**Коды, использовавшиеся в телеграфии**

В странах Восточной Азии, как и в США и в Европе, первые коды для представления символов национальных языков были разработаны для использования в телеграфии. Некоторые могут спросить, зачем они создавались специально, ведь существуют системы "латинизации" (представления иероглифов с помощью символов латинского алфавита, используемых для описания звуковых эквивалентов иероглифов - *прим. перев.*), позволяющие посылать данные, используя код Морзе?

На самом деле, "латинизация" не позволяет точно передать названия мест и личные имена, написанные китайскими иероглифами, так как существует слишком много омонимов (т.е. слов, которые пишутся по-разному, а звучат одинаково - *прим. перев.*). Поэтому и понадобилось пронумеровать тысячи символов китайского языка. Кроме того, были и другие соображения. Во времена расцвета телеграфа в Восточной Азии существовало только несколько публичных учебных заведений, и поэтому большинство получателей "латинизированной" телеграммы просто не смогли бы её прочесть.

Кроме того, в своё время посылка телеграфных сообщений стоила денег ( кое-где это до сих пор так... - *прим. перев.*), и притом очень больших. А поскольку использование "латинизации" влекло за собой удлинение сообщения, стоимость его посылки возрастала соответствующим образом. Случалось так, что в начале XX-го века стоимость посылки телеграфных сообщений была так высока, что даже в США и в Европе создавались специальные корпоративные и/или промышленные телеграфные коды для соединения нескольких слов в одно для того, чтобы экономить деньги при посылке сообщений\* ([здесь](http://www.research.att.com/~reeds/codebooks.html) находится более подробная информация на эту тему).

|  |
| --- |
| \* По этому поводу в России даже существует анекдот, правда, более поздний, где студент, не имеющий денег на посылку телеграммы из нескольких слов, передаёт родителям: "пятидесятирублируйте" - *прим. перев.* |

В результате символы языков стран Восточной Азии были впервые пронумерованы и разделены на наборы ещё во времена телеграфа. Тем не менее, наиболее важными кодами для представления символов этих языков были те, что были разработаны для использования в компьютерных системах в 1970-х, 1980-х и 1990-х гг., поскольку с тех пор именно они находятся в широком использовании.

**Коды для представления символов японского языка, используемые в компьютерах**

Европейцы были не последними людьми, которые базировали коды для представления символов своих национальных языков на рекомендации ISO 646. На другой стороне земного шара, в Японии, японская электронная индустрия была в процессе превращения себя в мировую силу, и одной из важнейших задач, на решении которой она сфокусировалась, была разработка компьютерных технологий, которые позволили бы работать с текстовыми данными, подготовленными на японском языке. Для достижения этой цели Японский Комитет по Промышленным Стандартам (Japan Industrial Standards Committee, JISC - аналог русского ГОСТ) поставил задачу утверждения таблиц символов для обеспечения информационного обмена на японском языке. Заметим, однако, что обеспечение работы компьютера с символами японского языка является значительно более сложной задачей, чем адаптация компьютера к работе с текстами на английском и других европейских языках.

В то время как для представления символов английского и других европейских языков достаточно использования 7-битных и 8-битных кодов, для японского языка использование 16-битного кода является обязательным условием. Это потому, что в японском языке используются четыре вида письменности: две слоговые (силлабические) "азбуки" *kana* - *katakana* и *hiragana* (далее в тексте такие термины на русский по ряду причин не переводятся - *прим. перев.*), несколько тысяч символов *kanji* (символы, заимствованные из китайского языка) и латиница. Несмотря на это, в 1970-х был создан 8-битный код JIS, предназначенный для работы с японским языком в "усечённом виде". Он назывался JIS C 6220 и в 1987-м году был переименован в JIS X 0201-1976.

Первая из 7-битных "половинок" его таблицы символов соответствовала "адаптированному для Японии" (полагаю, кавычки здесь уместны, см. ниже, почему - *прим. перев.*) национальному варианту ISO 646 (в нём место косой черты - "backslash " - было занято символом йены, и на месте тильды "~ " был расположен знак надчёркивания), а вторая была заполнена символами японских знаков препинания (например, начертание точки в японском языке отличается от начертания, принятого в европейских языках - *прим. перев.*) и символами слоговой азбуки *katakana*. Это позволяло осуществлять обмен текстовыми данными в ранних японских компьютерных системах. Таким образом, JIS X 0201-1976 включал в себя (в скобках указывается количество символов):

**JIS X 0201-1976**

|

+--символы цифр (10)

|

+--символы латинского алфавита (52)

|

+--специальные символы (32)

|

+--непечатаемые символы (34)

|

+--символы слоговой азбуки *katakana* (63 полуширинных символа, т.е.,

изображение одного иероглифа составлялось из двух последовательно

печатаемых символов; таким образом, каждый целый символ кодировался

с помощью 16-ти бит).

Конечно, поскольку первый японский код для представления символов японского языка не включал в себя даже слоговой азбуки *hiragana* (которая используется для написания собственно японских слов, в то время как с помощью *katakana* пишутся слова, заимствованные из других языков - *прим. перев.*), каждый, кто хотел легко и просто "сделать деньги" (на продаже компьютерной техники в Японии), должен был сделать ставку на то, что JISC должен будет скоро разработать эквивалент стандарта ISO 2022 для японского языка, который позволял бы переключатся между несколькими 8-битными таблицами символов. И JISC сделал это, назвав соответствующий стандарт JIS X 0202, который был опубликован на двух языках - японском и английском.

Затем JISC приступил к разработке таблиц символов (94x94) для большого количества символов *kanji*, используемых для письма в японском языке. В стандартах JIS символы *kanji* распределены по "уровням", хотя были среди них и такие, которые попали в число "вспомогательных и дополнительных *kanji* стандарта JIS", определённых отдельно. Первый из вышеуказанных "уровней" появился в стандарте JIS C 6226-1978, или т.н. "старом JIS". Позднее "старый JIS" переопределили, чтобы включить в него Jooyo kanji [символы *kanji* для повседневного использования], перечень которых был утверждён японским правительством в 1981-м году; это привело к установлению стандарта JIS 6226-1983, или т.н. "нового JIS". "Новый JIS" впоследствии был переименован в JIS X 0208-1983 в 1987-м году. И ещё позднее, в 1990-м, к *kanji* 1-го "уровня" были добавлены *kanji* 2-го "уровня", так появился стандарт JIS X 0208-1990, который сейчас является наиболее широко используемым. В дополнение к перечисленным, был выпущен ещё стандарт для представления дополнительных символов *kanji*, JIS X 0212-1990, однако в настоящее время его поддержка обеспечивается только в одной ОС для ПК (ОС, соответствующей спецификации BTRON, см. ниже).

Перечень символов, вошедших в эти два стандарта 1990-го года:

**JIS X 0208-1990**

|

+--знаки препинания, разные символы (93, 53)

|

+--алфавитно-цифровые символы, определённые в ISO 646 (10 цифр, 52 символа

| латиницы)

|

+--*hiragana* (83)

|

+--*katakana* (86)

|

+--греческий алфавит (48 - по 24 заглавные и строчные буквы)

|

+--современная кириллица (русский алфавит\*) (66 - по 33 заглавные и строчные

| буквы)

|

+--символы для рисования линий (32)

|

+--*kanji* 1-го уровня (2 965 символов, упорядоченных в соответствии с

| китайским стилем чтения)

|

+--*kanji* 2-го уровня (3 390 символов, упорядоченных в соответствии с

| корнями соответствующих китайских слов)

|

+--различные *kanji* (6 символов)

**JIS X 0212-1990**

|

+--диакритические знаки и символы (21)

|

+--греческие символы (21 буква с диакритическими знаками)

|

+--символы, используемые в языках стран Восточной Европы (26)

|

+--алфавитные символы (198)

|

+--*kanji* (5 801 символов)

|  |
| --- |
| \* Оцените широту японской души - с учётом того факта, что в русских ГОСТах японскими иероглифами и не пахло. Замечу попутно, что именно "благодаря" этим самым ГОСТам, многочисленным и противоречащим самим себе, в России стало невозможно нормально пользоваться даже собственными, русскими, буквами... - *прим. перев.* |

Как видно из вышеприведённого перечня, JIS X 0208-1990 является кодом для представления символов многих языков. В своём оригинальном виде стандарт JIS предполагал использование специальных escape-последовательностей - которые могли быть очень сложными - для переключения между символами ISO 646, для представления которых используется 7 бит, и символами *kanji*, для представления которых используется 16 бит. Однако поскольку всё это требует дополнительных затрат времени при работе с текстом и "съедает" много места на носителе информации, стандарт JIS в оригинальном виде был, надо понимать, неподходящим вариантом для использования в небольших компьютерных системах, таких как ранние 16-битные ПК.

Чтобы решить эту проблему, корпорация Microsoft изобрела "метод кодирования" для символов, описанных в стандартах JIS, названный Shift-JIS (или просто SJIS - *прим. перев.*) В нём нет escape-последовательностей и, соответственно, нет необходимости в переключении между частями этой таблицы. Это реализовано с помощью указания диапазона значения для 8-битных (алфавитных) символов и 1-го байта 16-битных (*kanji*) символов. В зависимости от значения, которое он находит в данных, кодирующих символ, компьютер определяет, столкнулся ли он с кодируемым 8-ю битами алфавитным символом или же с первым байтом кодируемого 16-ю битами символом *kanji*, и отображает их необходимым образом.

У этого метода есть два серьёзных недостатка. Первый состоит в том, что утверждённые Microsoft границы диапазонов не позволяют использовать все символы, определённые в стандартах JIS. Как следствие, компьютеры, в которых используется Shift-JIS, могут работать только с символами, описанными в стандарте JIS X 0208-1990, и не поддерживают *kanji*, описанными в стандарте JIS X 0212-1990 (так как они не вошли в фиксированную таблицу символов кода Shift-JIS). Второй недостаток - это то, что без использования специального ПО компьютер, в котором используется Shift-JIS, может работать только с такими текстовыми данными, в которых символы представлены с помощью 8-битных и 16-битных кодов. В таких средствах коммуникации, как Internet, эти данные в ряде случаев кодируются с помощью 7-битных кодов, что приводит к их искажению (разрушению) после загрузки.

По причине того, что Shift-JIS, как мы убедились, является методом (кодирования символов, описанных в стандартах JIS), весьма далёким от идеального, особенно когда он используется в такой среде, как Internet, для кодирования описанных в стандартах JIS символов были разработаны другие методы. Одним из них является код ISO-2022-JP (подмножество 7-битного кода ISO 2022), который широко используется в 7-битных системах Internet, для посылки электронной почты и сообщений в группы новостей. Как ясно из его названия, он базируется на описанном выше стандарте ISO 2022, и поэтому для переключения между символами ASCII и *kanji* используются escape-последовательности, упразднённые в Shift-JIS. И несмотря на то, что операторы (японских - *прим. перев.*) Internet-серверов предпочитают использовать ISO-2022-JP, они сообщают, что существуют, тем не менее, определённые проблемы с его использованием в языке HTML, на котором пишутся гипертексовые страницы.

Другим методом кодирования символов, описанных в стандартах JIS, является метод, получивший название UNIX Code-JP (EUC-JP). Из этого названия понятно, что он используется на компьютерах с Unix-совместимыми ОС, составляющих "хребет" сети Internet. EUC работает в 8-битных системах и представляет из себя простую схему, при которой код для представления символов языков государств Восточной Азии (китайского, японского и корейского) базируется на используемой в компьютере версии кода ASCII. В EUC это реализовано с помощью использования добавочных значений для распознования символов языков стран Восточной Азии. В случае EUC-JP, например, специальные добавочные значения добавляются к номерам символов, описанных в стандартах JIS, и это позволяет отличать их от символов кода ASCII.

Преимуществом этого метода является то, что возможно одновременное применение двух из четырёх таблиц символов (т.е., таблицы символов ASCII и одной из "национальных" таблиц - китайской, японской или корейской) без использования escape-последовательностей. Кроме того, EUC-JP позволяет (в отличие от Microsoft'оского Shift-JIS - *прим. перев.*) использовать *kanji*, описанные в стандарте JIS X 0212-1990. Недостатком является то обстоятельство, что EUC-JP требует для своего применения наличия мощного процессора и большого количества свободного места на носителе информации. Тем не менее, поскольку эти аппаратные ресурсы доступны в мире компьютеров, работающих на базе Unix-совместимых ОС, недостатки этого метода, понятно, не перевешивают его достоинств.

Пока JISC разрабатывал таблицы символов и методы кодирования - а он, по сути, продолжает делать это и сейчас; JISC открыл для пересмотра японской общественностью 3-й и 4-й "уровни" JIS - Китай, Корея и Тайвань следили за тем, что происходит, и, вслед за наиболее технически "продвинутым" из всех государств Восточной Азии (т.е. Японией), "компьютеризировали" свои национальные языки. Поэтому нет ничего удивительного в том, что таблицы символов, используемые в этих странах для кодирования символов, используемых в их национальных языках, - заметим, что во всех этих языках используются, наряду с собственными, символы из китайского языка - имеют довольно много общего с теми, что описаны в стандартах JIS. Например, как и стандарты JIS, они:

* базируются на "локализованных вариантах" стандарта ISO 646, в которых на месте символа "обратная косая черта" находится символ местной денежной единицы;
* построены с использованием таблиц 94x94;
* включают в себя символы алфавитов других языков (греческого алфавита, кириллицы, символы для представления иероглифов из японских слоговых азбук и т.д.);
* включают в себя элементы для рисования линий

и имеют другие общие черты. Тем не менее, китайская, корейская и тайваньская письменности имеют и собственные особенности. Ниже показано, какие символы включены в существующие в этих странах стандарты на таблицы символов.

**Коды для представления символов китайского языка, используемые в компьютерах**

Китайский национальный алфавит выделяется среди алфавитов государств Восточной Азии тем, что в нём используется самое большое количество символов "упрощённого китайского языка", и по этой причине таблица символов, используемая в КНР, называется "Simplified Chinese" ("Упрощённая китайская"). Символы "упрощённого китайского языка" изначально предполагалось использовать для создания простой для изучения системы "латинизации" национального письма, которая позволила бы увеличить производство национальной литературы в Китае. Однако эта задача была отменена из-за проблемы огромного числа возникающих при этом омонимов. В результате набор символов китайского языка, используемый в Китае, существенно отличается от такового, применяемого на Тайване.

В настоящее время в Китае действует стандарт GB 2312-80 (аббревиатура GB происходит от "*Guo-jia Biao-zhun*", что означает "Национальный стандарт"). В него включены следующие символы:

**GB 2312-80**

|

+--различные символы (94)

|

+--числительные (72)

|

+--ISO 646-CN (94 полноширинных символа)

|

+--японская *hiragana* (83)

|

+--японская *katakana* (86)

|

+--греческий алфавит (48)

|

+--кириллица (66)

|

+--символы *pinyin* и *bopomofo* (26, 37)

|

+--элементы для рисования линий (76)

|

+--*hanzi*\* 1-го уровня (3 755, упорядоченные в соответствии с правилами

| чтения *pinyin*)

|

+--*hanzi* 2-го уровня (3 008, упорядоченные в соответствии сначала с

корнями китайских слов, а затем - в соответствии со штрихами, образующими

символ).

|  |
| --- |
| \* Очевидно, "*hanzi*" означает то же, что и "*kanji*". - *прим. ред.* |

Для кодирования символов из таблицы "Simplified Chinese" используются следующие методы: 7-битный ISO 2022, ISO-2022-CN (для сообщений электронной почты), EUC-CN и HZ (HZ-GB-2312). Метод кодирования HZ используется для сообщений электронной почты и был создан китайским исследователем в Стэнфордском университете в США. Он аналогичен 7-битному ISO 2022; главное отличие состоит в том, что в HZ для переключения с 8-битных (алфавитных) символов на 16-битные (*hanzi*) используются не escape-последовательности, а shift-последовательности, состоящие из двух печатаемых символов.

**Коды для представления символов корейского языка, используемые в компьютерах**

Корея - это единственная страна в Восточной Азии, национальная письменность которой имеет свой алфавит, он называется *hangul*. Несмотря на то, что в письме вместе с символами *hangul* используются и символы из китайского языка, в Корее существует тенденция по возможности отказываться от использования китайских символов, за исключением тех, что используются для названий мест и личных имён. Корейский алфавит (*hangul*) содержит всего 24 буквы (10 гласных и 14 согласных), поэтому, казалось бы, работу с корейским языком в компьютере реализовать не так сложно. Однако в письме буквы корейского алфавита образуют слоговые блоки, в которых символы складываются вертикальной "стопкой", один поверх другого, и это привело к жарким спорам вокруг того, как именно символы простой корейской системы письменности должны обрабатываться в компьютерах. Действующий в Корее стандарт для обработки данных и обмена информацией, KS C 5601-1992 (KS - сокращение от "Korean Standard"), включает в себя наиболее часто используемые слоги в виде заранее предопределённых слоговых блоков. Он "устроен" следующим образом:

**KS C 5601-1992**

|

+--символы (94)

|

+--аббревиатуры и символы (69)

|

+--ISO 646-KR (94 полноширинных символа)

|

+--элементы *hangul* (94)

|

+--римские цифры и буквы греческого алфавита (68)

|

+--элементы для рисования линий (68)

|

+--аббревиатуры (79)

|

+--фонетические символы, символы в кружках и дроби (91)

|

+--фонетические символы, символы в круглых скобках, подстрочные и

| надстрочные символы (94)

|

+--*hiragana* (83)

|

+--*katakana* (86)

|

+--кириллица (66)

|

+--*hangul* (2 350 заранее составленных слоговых блоков)

|

+--*hanja* (4 888 символов китайского языка, упорядоченных в

соответствии с *hangul*, а затем - по корням слов)

Для кодирования символов корейского языка используются следующие методы: 7-битный ISO 2022, ISO-2022-KR (для сообщений электронной почты), EUC-KR и *Johab*. *Johab*, или "код из двухбайтных комбинаций", представляет из себя систему, в которой представлены все возможные слоговые блоки (в общей сложности их 11 172), вне зависимости от того, встречаются они реально в корейском языке или нет.

Корпорация Microsoft создала также код UHC ("Unified Hangul Code"), называемый ещё "Extended Wansung", для корейской версии своей ОС Windows 95 (Win95K), который обеспечивает совместимость со стандартом KS C 5601-1992 при добавлении поддержки системы кодирования *Johab* (8 822-х заранее определённых слоговых блоков, которых нет в стандарте KS C 5601-1992). Дополнительно корпорация Microsoft планирует поддерживать эту систему в корейской версии своей ОС Windows NT.

**Коды для представления символов тайваньского языка, используемые в компьютерах**

Несмотря на то, что и в китайском, и в тайваньском языках для разговорной речи используется один и тот же так называемый "мандаринский китайский язык", письменность Тайваня отличается от китайской - она базирована на виде китайских символов, который использовался на протяжении веков, поэтому тайваньский набор символов называется "Traditional Chinese " ("традиционный китайский"). Китайские символы из таблицы "Traditional Chinese" по чистой случайности очень схожи с теми китайскими символами, что используются в Гонконге и Сингапуре.

На данный момент существует две таблицы символов, используемые для компьютерной обработки текстов, написанных на "традиционно китайском". Это Big-5 (названа так потому, что она была принята пятью крупнейшими производителями компьютерной техники) и CNS 11643-1992 (CNS - сокращение от "Chinese National Standard"), которые совместимы между собой лишь частично. Главное отличие состоит в том, что таблица CNS 11643-1992 содержит в 3,5 раза больше символов, чем таблица Big-5. Вот как выглядят эти таблицы:

**Big-5**

|

+--символы (157)

|

+--символы (157)

|

+--символы (94)

|

+--*hanzi* 1-го уровня (5 401 символ китайского языка. Упорядочены по

| количеству штрихов, затем - по корням соответствующих слов)

|

+--*hanzi* 2-го уровня (7 652 символа китайского языка. Упорядочены по

количеству штрихов, затем - по корням соответствующих слов.

(Замечание: 2 китайских символа в *hanzi* 2-го уровня повторяют уже

имеющиеся, таким образом, в этой таблице только 7 650 уникальных

символов *hanzi*).

**CNS 11643-1992** (только символы и *hanzi*):

|

+--символы (438)

|

+--классические корни слов (213)

|

+--графические представления управляющих символов (33)

|

+--*hanzi* 1 (5 401 символ китайского языка)

|

+--*hanzi* 2 (7 650 символов китайского языка)

|

+--*hanzi* 3 (6 148 символов китайского языка)

|

+--*hanzi* 4 (7 298 символов китайского языка)

|

+--*hanzi* 5 (8 603 символа китайского языка)

|

+--*hanzi* 6 (6 388 символов китайского языка)

|

+--*hanzi* 7 (6 539 символов китайского языка)

Для кодирования символов "традиционного китайского" языка используются следующие методы: 7-битный ISO 2022, ISO-2022-CN (для e-mail сообщений), EUC-TW и Big-5. Таблица символов и метод кодирования Big-5 используются в китайских версиях ОС Microsoft MS-DOS и Windows и ОС компьютеров Macintosh, производимых корпорацией Apple.

Необходимо отметить, что все перечисленные выше в этом разделе таблицы символов обновлялись, обычно с помощью расширения, несколько раз. Когда такие обновления делаются в логической и последовательной манере, пользователи компьютерных систем, конечно, от этого выигрывают, так как получают возможность использовать более полный набор символов своего языка. Однако если обновления приводят к сдвигу в таблице позиций с ранее использовавшимися символами, обмен информацией между системами, использующими разные версии даже одной и той же таблицы символов, может привести к потере данных. В частности, это имело место с таблицами, используемыми для символов, описанных в стандартах, выпущенных JISC.

## Таблицы символов и методы кодирования, применяемые для работы с многоязычными текстами

**Ранние многоязычные таблицы символов и методы кодирования**

Изначально таблицы символов применялись только для удовлетворения нужд по обмену информацией и редактированию текстов на национальных языках конкретных государств, однако в реальном мире нередко бывает необходимо использовать в одном документе символы нескольких разных языков одновременно. Так, книгоиздательской промышленности нужно использовать множество символов иностранных языков при выпуске словарей иностранных слов или, например, книг иностранных авторов на языке оригинала. А с тех пор, как появилась сеть Internet, похоже, каждый человек по мере работы в ней будет сталкиваться с данными на иностранном языке, написанными набором символов, используемых другими нациями.

По этой причине эффективные многоязычные таблицы символов и методы для кодирования этих символов совершенно необходимы для жизни в XXI-м веке, но проблема в том, что они слишком долго не являлись предметом научных исследований и разработок. Самые первые такие таблицы и методы кодирования появились в первой половине 1980-х, в те времена, когда большинство компьютеров были 8-битными и 16-битными.

Наиболее достойными внимания среди них были рабочие станции Star, производимые корпорацией Xerox и имеющие программу для обработки многоязычных текстов, называемую ViewPoint, а также офисные компьютеры модели 5550, производимые корпорацией IBM. Обе эти системы позволяли работать с большим количеством азиатских языков, в дополнение к большому количеству языков, использующих латинский алфавит, но они так никогда и не вошли в широкое употребление из-за дороговизны. Однако многоязычная таблица символов, предложенная корпорацией Xerox, была взята на вооружение специалистами по компьютерной технике и лингвистами в США. Это в конце концов привело к созданию движения Unicode, предложившего свою многоязычную таблицу символов (и методы кодирования этих символов), которая сейчас является одним из главных соперников в борьбе за международное признание (см. ниже).

**Проект TRON - многоязычная таблица символов и многоязычное окружение**

Это может кого-то удивить, но Япония, которую часто (и совершенно несправедливо - *прим. перев.*) критикуют за закрытость и ретроградные настроения в обществе, также предложила собственную многоязычную таблицу символов и метод для кодирования этих символов ещё в первой половине 1980-х. Это было сделано в рамках проекта TRON, который был начат в 1984-м. Задачей проекта TRON является создание тотальной, открытой компьютерной архитектуры, базированной на используемой в настоящее время архитектуре фон Неймана\*, для компьютеризации человеческого сообщества в XXI-м веке. Безусловно, компьютеризация человеческого сообщества влечёт за собой преобразование огромной массы человеческих знаний таким образом, чтобы они стали доступны каждому в режиме реального времени (например, через Internet), а это требует создания хорошо спланированной таблицы символов естественных языков и эффективной системы для работы с многоязычными данными на ПК и рабочих станциях.

|  |
| --- |
| \* Это всё звучит хорошо, но большинство специалистов всё же не разделяет оптимистичных настроений автора по данному поводу - см., например, ["Компьютерные архитектуры: неожиданные повороты"](http://consumer.nm.ru/p2_z1.htm) и ["Мегабитовая бомба"](http://msk.nestor.minsk.by/kg/2001/48/kg14806.html). - *прим. перев.* |

На данный момент разработаны открытая для расширения таблица символов TRON и многоязычное окружение TRON, являющиеся частью соответствующей спецификации операционной системы BTRON. На данный момент права на эту систему куплены корпорацией Personal Media. Последняя версия этой ОС, B/V (используется правая косая черта), выпускаемой Personal Media, обеспечивает одновременную работу с символами нескольких языков Восточной Азии в качестве стандартной возможности.

В следующей версии этой ОС, имеющей предварительное название "3B/V", будет возможно использовать полный, без каких-либо сокращений, набор *kanji*, состоящий приблизительно из 65 000 символов. Он назван GT Micho и в настоящее время проходит утверждение в Университете Токио.

Таблица символов и метод кодирования TRON, а также многоязычное окружение TRON были впервые описаны на английском языке во время проведения 3-го симпозиума по проекту TRON в 1987-м году. В применённом в TRON подходе к обработке многоязычных документов существует ряд особенностей, делающих его уникальным. Одна из них состоит в том, что таблица символов TRON расширяема практически до бесконечности, что позволяет включить в нес символы всех без исключения языков, когда-либо использовавшихся (и используемых сейчас) человечеством, и даже символы тех языков, которые ещё не изобретены. Это реализовано с помощью escape-последовательностей, которые используются для переключения между очень большим количеством ("внутренних") таблиц, которые содержат символы, кодируемые с помощью 8-ми и 16-ти бит. Кроме того, в TRON имеются коды для идентификации языка, которые необходимы, например, для возможности реализации программ для правильной сортировки текстовых данных, содержащих символы разных языков.

Среди других преимуществ - то, что в TRON нет места для символов, определяемых пользователями, которые (символы) могут вызывать проблемы при обмене данными. Кроме того, в TRON проводится чёткое разделение между текстовыми данными, предназначенными для хранения на носителе информации, и данными, предназначенными для отображения, поэтому, например, два символа данных, хранимых на носителе информации, при выводе на дисплей могут быть объединены в один, так чтобы образовывать лигатуру. Среди особенностей также: существование специальной базы данных по таблице символов, которая особенно необходима для правильного использования огромного количества символов GT Mincho, и система для многоязычного письма, которая позволяет передавать многоязычные данные через WWW и изображать их с приемлемым качеством, при этом без необходимости в применении сложных алгоритмов для вёрстки.

Соответствующая спецификации BTRON3 операционная система B/V, в которой поддержка многоязычного окружения TRON, заметим, ещё полностью не реализована, в настоящее время поддерживает работу со следующими таблицами символов:

**Таблицы символов BTRON3 (частично)**

|

+--таблица ISO 8859-1

|

+--таблица из стандарта GB 2312-80

|

+--таблица из стандарта JIS X 0208-1990

|

+--таблица из стандарта JIS X 0212-1990

|

+--таблица из стандарта KS C 5601-1992

|

+--6-точечная азбука Брайля

|

+--8-точечная азбука Брайля

Несмотря на то, что на данный момент таблица символов TRON и многоязычное окружение TRON превосходят все другие системы, предложенные для обработки данных и обмена информацией, многие будут настаивать, что у TRON слишком мало шансов стать стандартом, поскольку ряд ведущих американских корпораций-производителей ПО, выпускающих операционные системы, противостоит и препятствует этому. Однако сторонники этой точки зрения упускают из виду тот очевидный факт, что у японского общества существует реальная практическая необходимость в такой ОС для ПК, которая поддерживала бы работу со всеми существующими на данный момент символами *kanji*. Кому это надо? Начинающим пользователям, банкам, государственным учреждениям, школам и многим другим организациям, которым необходимо заносить имена японцев в списки и базы данных в их оригинальном виде.

Более того, японским библиотекам необходимо компьютеризировать их карточные каталоги и размещать в свободном доступе в сети (например, в Internet) те части своих коллекций, срок авторских прав на которые исчерпан. Наконец, поскольку ПО для всрстки разрабатывается для использования на базе ОС, совместимой со спецификацией BTRON, TRON нужен японской книгоиздательской промышленности. Поэтому всё не так уж сумрачно, - даже если TRON и не снискал "благословения" от ряда американских корпораций, желающих ещё долго единолично контролировать сферу производства ОС, - и таблица символов TRON, а также многоязычное окружение TRON, имеют все шансы на долгое применение.

**Unicode и ISO 10646**

Как уже было сказано выше, американские компьютерные корпорации начали в первой половине 1980-х гг. работу над созданием многоязычной таблицы символов и методов для кодирования этих символов. Результаты исследовательских работ в данной области, проведённых корпорациями Xerox и IBM, были успешно воплощены этими корпорациями в своих компьютерных системах. Исследователи из Xerox в последующем "обратили в свою веру" представителей других компьютерных корпораций США, и в конце концов они вместе начали проект американской индустрии, названный Unification Code, или Unicode, главной задачей которого было сведение всех существующих в мире таблиц символов в единую таблицу символов.

Метод Unicode также должен был быть несложным для обработки на компьютерах, и до настоящего времени имел две важных (приоритетных) черты:

1. избежание использования escape-последовательностей для переключения между ("внутренними") таблицами символов и
2. ограничение количества используемых символов: каждый из них представляется с помощью двоичных последовательностей, имеющих строго фиксированную длину - 16 бит, что позволяет использовать максимум 65 536 символов (2 в 16-й степени).

Несмотря на всё это, ISO, создатель этих надоевших escape-последовательностей, декларированных в широко используемом стандарте ISO 2022, также хотела создать (свои) многоязычную таблицу символов и метод для кодирования этих символов. В отличие от Unicode, целью ISO было создание универсальной 32-битной таблицы символов UCS (Universal Coded Character Set), в котором предполагается использовать escape-последовательности для переключения между ("внутренними") таблицами символов. 32-битная таблица символов имеет достаточно позиций, чтобы обеспечить вмещение примерно 4,3 миллиардов символов (2 в 32-й степени = 4 294 967 296). Иными словами, она практически бесконечна и пригодна для любого практического применения\*.

|  |
| --- |
| \* В отличие от ранних наборов символов, в которых символы упорядочивались в маленьких таблицах с сеткой 16x16 или 94x94 ряда, в Unicode и ISO UCS используются таблицы символов с большой сеткой 256x256. Главное различие между ними состоит в том, что Unicode использует одну такую сетку (что и даст 65 536 символов - *прим. перев.*), а UCS - тысячи их. - *прим. авт.* |

Этот стандарт ISO на многоязычную таблицу символов, получивший название ISO/IEC DIS 10646 Version 1 (чаще называемый просто ISO/IEC 10646-1 - *прим. перев.*), был поддержан японскими и европейскими исследователями. Однако он не был поддержан американскими корпорациями-производителями компьютерной техники и ПО, которые параллельно вели разработку стандарта Unicode, и зашли в этом даже так далеко, что создали "консорциум Unicode" для проведения своих исследований. Они считают, что Unicode лучше стандарта ISO/IEC DIS 10646 Ver. 1 просто потому, что он проще.

Поэтому американские корпорации-члены консорциума Unicode, в ответ, предложили ISO сделать Unicode базовой "внутренней" таблицей\* для многоязычной таблицы символов ISO. Конечно, поскольку они - те, кто они есть, и являются в том числе и производителями наиболее широко используемых ОС в мире, они в состоянии создать свою альтернативу практически любой системе, которую могла бы разработать ISO. Соответственно, им удалось убедить ISO в том, что стандарт ISO/IEC DIS 10646 Version 1 лучше отвергнуть. Это было сделано - появился стандарт ISO/IEC 10646 Version 2, базовой многоязычной таблицей символов в котором объявлялась таблица, используемая в Unicode. По существу, Unicode поглотил стандарт ISO (на самом деле, конечно, наоборот, Unicode "проник" в него, или даже можно сказать "по сути, стал стандартом ISO..." - *прим. перев.*), называемый теперь ISO/IEC 10646-1: 1993. Этот стандарт "устроен" следующим образом:

**ISO/IEC 10646-1: 1993**

|

+--ISO 646

|

+--ISO 8859-1

|

+--символы с ударениями и диакритическими знаками, используемые в языках

| Восточной Европы

|

+--Международный фонетический алфавит (International Phonetic Alphabet, IPA)

|

+--символы греческого языка (включая символы с ударениями, *monotoniko* и

| *polytoniko*)

|

+--кириллица, грузинский и армянский алфавиты

|

+--символы иврита

|

+--арабские буквы (во всех четырёх видах: начинающих слово, серединных,

| оканчивающих слово и отдельных)

|

+--символы алфавитов индийских языков (включая *Devanagari, Bengali, Gurmukhi,*

*| Gujarati, Oriya, Tamil, Telugu, Kannada и Malayalam*)

|

+--символы тайского и лаосского языков

|

+--идеографические символы CJK - китайского, японского и корейского языков

| (включая *hangul*, *katakana*, *hiragana* и *bopomofo*)

|

+--математические операторы и символы специальной формы

|

+--символы для рисования квадратов и линий

|

+--геометрические фигуры и идеограммы

|

+--Специальные оптически распознаваемые символы, используемые в чеках

|

+--символы и цифры в кружках

|  |
| --- |
| \* Следует отметить, что базовая "внутренняя" таблица символов (Basic Multilingual Plane) многоязычной таблицы символов ISO в оригинальной версии стандарта ISO/IEC DIS 10646 была описана следующим образом: "Базовая "внутренняя" таблица: "внутренняя" таблица, содержащая широко применяемые неидеографические символы". Иными словами, она никогда не предназначалась быть стандартом ISO для символов китайского, японского и корейского языков (что и естественно, поскольку для всех их сразу она была слишком мала; как мы помним, объём одной "внутренней" таблицы, в том числе и базовой, равен 65 536 символам - *прим. перев.*). - *прим. авт.* |

Внимательный читатель может теперь спросить, каким же образом "все символы мира" могут уместиться в 65 536 позициях (предлагаемых как Unicode, так и существующим на данный момент стандартом ISO на многоязычные таблицы символов - *прим. перев.*), если выше было упомянуто, что в Университете Токио разрабатывается набор из примерно 65 000 символов только для японского языка!? Ответ состоит в том, что это достигается через "унификацию" похожих друг на друга (по мнению консорциума Unicode - *прим. перев.*) символов, используемых в этих языках. Нет, все буквы "А" в таблицах с символами алфавитов европейских языков не "унифицируются" при этом в одну позицию... Что было "унифицировано", так это тысячи символов, составляющих письменность языков государств Восточной Азии.

Консорциумом Unicode была для этого создана "объединённая исследовательская группа по китайскому, японскому и корейскому языкам" (Chinese/Japanese/Korean Joint Research Group, CJK-JRG), которая и сейчас загружена выполнением главной задачи, поставленной консорциумом: "унификации к Хань" - *Han Unification*. (С учётом того, что "Хань" - это самоназвание китайской нации, суть данного предприятия, очевидно, состоит в том, что "основным" начертанием символа, которое при "унификации" берётся за эталонное и вносится в таблицу, считается начертание этого символа, принятое в китайском языке - *прим. перев.*), даже несмотря на то, что подавляющее большинство людей, представляющих культурную сферу Хань, вероятно, просто не отдают себе отчёта в том, что они делают. В настоящее время они определили в многоязыковой таблице символов местá для примерно 20 000 символов китайского языка, и им остастся проделать эту работу для ещё примерно 30 000 символов.

Несмотря того, что система Unicode уже неоднократно была раскритикована как нечто даже большее, чем проявление культурного империализма рядом западных компьютерных корпораций, на самом деле наиболее серьёзные проблемы Unicode носят технический характер и их можно перечислить:

1. в Unicode не предусмотрено информации для определения конкретного языка, к которому относятся символы, что влияет на порядок их сортировки и т.д.
2. допускается использование символов, определённых не стандартом, а произвольно конкретными пользователями, что вызывает проблемы при обмене информацией (между компьютерами, в таблицах символов которых эти символы определены по-разному - *прим. перев.*)
3. не предусматривается возможности дальнейшего расширения (т.е. Unicode не может перешагнуть 16-битный барьер - 65 536 символов - *прим. перев.*)
4. в Unicode слишком много похожих символов (см. выше про "неунифицированные" буквы "А" европейских языков - *прим. перев.*)
5. некоторые символы создаются с помощью наложения одного на другой, что делает невозможным их использование в моноширинных шрифтах
6. преобразование из/в Unicode не может простым образом осуществляться для текстов, в которых используются китайский, японский и корейский языки.

Даже несмотря на то, что одной из главных сил, стоящих за Unicode, является могучая корпорация Microsoft, применяющая Unicode в серверной версии своей ОС Windows NT, Unicode встречает ещё более мощное сопротивление в странах Восточной Азии. Это не только потому, что народы стран Восточной Азии не желают, чтобы американские корпорации решали за них, какие символы им можно использовать в представленных в электронном виде данных, а какие - нельзя (хотя, конечно, эта причина сама по себе многого стóит). Более важно то, что Unicode просто не отвечает требованиям, предъявляемым обществами стран Восточной Азии, особенно в перспективе.

К примеру, Unicode не подходит для создания доступных через Internet в режиме электронных библиотек. Такие библиотеки требуют использования полного, без каких-либо сокращений, набора символов, включающего каждый из символов, которые были когда-либо использованы. Unicode не позволяет этого сделать. Использующие Unicode компьютерные системы также не могут быть использованы для записи имён многих японских людей и названий многих японских местностей, и поэтому они не могут использоваться для компьютеризации государственных учреждений Японии. Ввиду ограничений, проистекающих из самой его природы, Unicode, аналогично, не может быть использован для компьютеризации магазинов, продающих печатную продукцию, и издательств в странах Восточной Азии. Поэтому, если разобраться, Unicode на самом деле является куда более выгодным для компьютеров и их производителей, а не пользователей, проживающих в странах Восточной Азии. С учётом описанных выше основных черт Unicode, это заключение даже и не выглядит чем-то удивительным.

Дополнительную информацию о консорциуме Unicode можно получить на его официальном [сетевом узле\*](http://www.unicode.org).

|  |
| --- |
| \* После того, как эта статья была написана, консорциум Unicode реализовал механизм, позволяющий системе кодирования Unicode обрабатывать более 1 миллиона симоволов. Ознакомиться с этим механизмом, а также с его оценкой проектом TRON Web, можно, прочтя статью ["Unicode Revisited"](http://tronweb.super-nova.co.jp/unicoderevisited.html), также написанную автором. - *прим. авт.* |

**UNIX и Mule**

Люди, создавшие сеть Internet, которая сейчас становится главным каналом для международного обмена многоязычными данными, были не кто иные, как программисты и инженеры, которые проводили день за днём за экранами своих рабочих станций, базированных на Unix-совместимых ОС. Их мир очень сильно отличается от мира персональных компьютеров. Они работают на высокопроизводительных рабочих станциях, операционные системы которых являются частично или полностью открытыми. Они имеют доступ к огромному количеству бесплатного ПО и средств программирования, и люди, которые пишут эти программы, часто делают их доступными для других членов UNIX-сообщества бесплатно.

Поэтому в отличие от мира персональных компьютеров, где технические стандарты создаются наиболее сильными рыночными лидерами, главной задачей которых является создание и поддержание технологической гегемонии, мир UNIX-совместимых ОС основывается на открытости, сотрудничестве и терпимости. Неудивительно, что в таком мире существует только небольшая поддержка Unicode, которая является при этом противоречивой и до сих пор остаётся до конца не реализованной. Тем не менее, кого-то может удивить тот факт, что применяемое в мире UNIX решение проблемы поддержки работы с многоязыковыми данными было разработано Японии, так же как и описанное выше многоязыковое окружение TRON.

Подход, использованный в UNIX-совместимых ОС для обработки многоязыковых данных, прост, хотя и требует сравнительно больших вычислительных мощностей для реализации. UNIX принимает "как есть" тексты, вне зависимости от того, с использованием какой таблицы символов они были подготовлены, и позволяет обрабатывать их с помощью специального ПО для обработки текстов (в том числе многоязыковых), в котором для работы с этими таблицами используется унифицированный и расширяемый механизм. Эта программа называется Mule (MULtilingual Enhancement to GNU Emacs) и была разработана в японской государственной электротехнической лаборатории научного городка Тускуба, который расположен в северо-восточной части Токио. Исследовательской группой, проделавшей эту работу, руководил Кеничи Хэнда (Kenichi Handa).

Программа Mule, которая была представлена как свободный продукт в 1993 году, "выросла" из ранее существовавшего усовершенствования текстового редактора GNU Emacs (свободное ПО), известного (речь идёт о "расширении" - *прим. перев.*) как Nemacs, или Nihongo [японский] Emacs. Как и в многоязыковом окружении TRON (и изначально предлагавшейся ISO системе UCS), в Mule для переключения между языками используются escape-последовательности. Существующая в настоящее время версия может работать с греческим, русским и европейскими языками, языками стран Восточной Азии, тайским, вьетнамским, арабским языками, с ивритом и другими. В настоящее время разрабатывается также поддержка работы с языками, используемыми на индийском полуострове. Вся эта система будет интегрирована в оригинальную версию текстового редактора GNU Emacs для распространения по всему миру.

Для того, чтобы получить более подробную информацию о Mule, щёлкните [здесь](http://www.etl.go.jp/~mule/).

Некоторые люди могут сказать что поскольку Mule был разработан профессиональными разработчиками ПО для UNIX-сообщества, у него мало шансов войти в широкое употребление. Однако имеется ряд факторов, которые могут привести к тому, что Mule вырвется из традиционного мира Unix-совместимых ОС и получит широкое признание. Во-первых, Mule позволяет людям использовать существующие таблицы символов и способы кодирования, поэтому он не является противоречивым. Во-вторых, ПК сейчас даже превосходят по вычислительным мощностям ранние рабочие станции, базированные на Unix-совместимых ОС, поэтому аппаратные ресурсы для использования Mule - имеются. В-третьих, Unix-совместимая свободная ОС Linux начинает получать широкое признание - особенно в Европе - поэтому люди могут без проблем перейти на эту платформу и использовать Mule.

Единственный вопрос в том, как много людей пожелает переключиться на использование Unix-совместимых ОС с ОС, используемых в настоящее время на ПК? С другой стороны, если смотреть на перспективы, можно предположить, как долго ещё Mule будет оставаться на сцене. Mule - это явно временная мера, которая удовлетворяет текущие потребности по работе с многоязычными данными, в то время как для многоязычного информационного обмена в XXI-м веке предлагаются конфликтующие стандарты. Тем не менее, до тех пор, пока один из этих стандартов не станет очевидным лидером, Mule, вероятно, будет представлять ценность для людей, которым необходимо использовать многоязыковое вычислительное окружение, и которые готовы ради этого сменить используемую ОС.

**Будущее многоязычных таблиц символов и способов для кодирования этих символов**

Как знает каждый, кто имеет хотя бы самое общее представление о мире компьютерной техники, будущее компьютеров начинает обращаться вокруг компьютерных сетей мирового масштаба. Сегодняшняя сеть Internet представляет из себя эту глобальную общемировую сеть пока в эмбриональном состоянии. Тем не менее, эти компьютерные сети мирового масштаба не смогут стать эффективными до тех пор, пока не будет проложено больше высокоскоростных линий связи, пока они не будут подключены к заводам, офисам и жилищам, и пока не будет создано новых стандартов, регламентирующих процессы обмена информацией. Среди прочих стандартов этого рода, одним из наиболее важных будет стандарт, определяющий многоязычную таблицу символов и способ для кодирования этих символов, используемые для обеспечения обработки данных на всех языках мира.

На данный момент для решения этой задачи предложено два стандарта. Один из них - это TRON, включающий в себя таблицу символов TRON и многоязычное окружение TRON, а другой - это Unicode. Тем временем многоязычный текстовый редактор Mule, работающий в UNIX и Unix-совместимых ОС, представляет собой временное решение.

Что произойдёт в будущем? Это сложно предсказать, но поскольку американские корпорации-поставщики компьютерной техники и ПО с помощью правительства США или без таковой пытаются внедрить Unicode в окружающий их мир своими традиционными методами, в виде открытия рыночной кампании, наиболее вероятным исходом будет битва между соперничающими стандартами, один из которых в конце концов будет решено использовать в качестве базового. И это будет тот стандарт, который, подчеркнём, выберут пользователи (а не производители) компьютерных систем.

Коммерчески доступная, базированная на BTRON операционная система работает на том же самом аппаратном обеспечении, что и базированная на Unicode Microsoft Windows NT, поэтому в Японии всем пользователям достаточно всего лишь переключиться с одной используемой на жёстком диске файловой системы на другую, для того, чтобы сделать выбор по своему усмотрению. Более того, поскольку протоколы Internet, как было сказано выше, позволяют использовать любые таблицы символов и методы кодирования символов, эта битва стандартов окажет лишь незначительное влияние на строение сетей передачи данных.

И поэтому можно с уверенностью сказать, что у пользователей персональных компьютеров, впервые за долгое время, реально появился выбор!